Robot Arm Manipulation using Online Behavior Cloning with Softer Actor-Critic

Keivalya Bhartendu Pandya

Northeastern University,

pandya.kei@northeastern.edu

Abstract

In this paper we present a three-phase Soft Actor-Critic (SAC) method that utilizes expert data, in the form of human demonstrations $\mathscr{D}$ to solve continuous-state-space, multi-stage, long-horizon robotic tasks. This general and universally-applicable approach is based on reducing the reliance on $\mathscr{D}$ in each learning phase, and repopulating the replay buffer with the experiences learned by the agent's trained policy. Our method outperforms Vanilla SAC, including as well as excluding $\mathscr{D}$, keeping other hyper-parameters unchanged across all studied methods. While this method relies on demonstration data to bootstrap relay buffer memory, we do not fully rely on it at any given point in the time of training the agent. We randomly sample a small ratio of data from $\mathscr{D}$, and initialize the policy with random behavior in the beginning. We demonstrate the effectiveness of our method on a number of long-horizon tasks in a challenging kitchen simulation environment. By incorporating an entropy-based reward mechanism, the algorithm promotes exploration, reducing the likelihood of premature convergence to suboptimal solutions. This capability aligns closely with our objective of designing an agent that efficiently transitions from relying on demonstration data to developing its policy through exploration, ensuring adaptability and robustness in complex environments.

Code — https://github.com/keivalya/arm-manipulation-behavior-cloning

Dataset — https://huggingface.co/datasets/keivalya/arm-manipulation-behavior-cloning

Introduction

Reinforcement Learning (RL) holds the promise of continually improving policies through online trial-and-error experiences (Sutton & Barto, 2018), making it an ideal choice for developing robots that can adapt to various environments. In the context of household robotics, where robotic arms are designed to perform more routine tasks such as operating kitchen appliances, cooking and food preparation, training RL agents often demands an impractically large number of samples, making it a costly and inefficient process. To address this, recent advances in robot learning have increasingly relied on behavior cloning (BC; Abhishek Gupta, 2016), which enables the development of effective policies by leveraging offline demonstrations from humans, reducing the need for extensive live training.

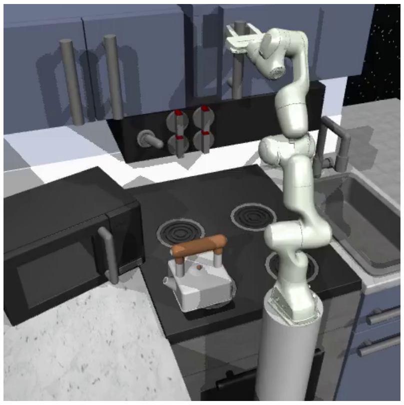

Figure 2. Simulation Environment

Despite the use of BC in leveraging expert demonstrations to accelerate learning, several challenges persist in robotic arm manipulation tasks. A critical question arises: how can we ensure that the agent effectively integrates the benefits of demonstration data while retaining the ability to explore and improve through its own experiences? Over-reliance on demonstrations risks limiting the agent's adaptability and can lead to suboptimal performance in unseen scenarios. On the other hand, insufficient use of demonstration data may result in inefficient learning, requiring excessive interaction with the environment. Striking a balance between these extremes is crucial for solving multi-stage, long-horizon tasks, which are inherently more complex due to their sequential dependencies and continuous action spaces.

Our proposed method addresses these challenges by incorporating a three-phase SAC approach with online behavior cloning. The approach introduces a strategic interplay between demonstration data and the agent's online exploration. By incrementally reducing reliance on expert demonstrations across the training phases, our method encourages the agent to develop and refine its own policy, guided but not constrained by the initial demonstrations. A key novelty lies in repopulating the replay buffer with experiences generated by the agent's trained policy, allowing the learning process to dynamically adapt and prioritize meaningful transitions. Additionally, the use of a small, randomly sampled ratio of demonstration data throughout training serves to bootstrap early learning without creating dependency.

Related Work

BC, a form of imitation learning, enables robots to acquire skills by mimicking expert demonstrations. However, BC often suffers from distribution shift, where deviations from the training data lead to compounding errors during execution. To address this issue, methods such as DAgger (Dataset Aggregation) have been developed, which iteratively refine the policy by incorporating the learner's experiences alongside the demonstrations (Ross et al., 2011). While effective, DAgger's repeated data collection cycles can make it computationally expensive for real-world applications.

A noteworthy contribution to this field is the work on Relay Policy Learning (RPL) by Gupta et al. (2020), which proposes a staged approach to reduce the reliance on demonstrations. In RPL, the agent incrementally learns from demonstration data, transitioning towards autonomous exploration by training intermediate policies that serve as stepping stones. This approach effectively mitigates distribution shift and demonstrates promising results in multi-stage tasks. However, RPL's staged nature can introduce challenges in determining the optimal transition points between stages, potentially leading to inefficient policy handoffs.

Another pertinent study, *What Matters in Learning from Offline Human Demonstrations for Robot Manipulation*, explores offline RL approaches to leverage human demonstrations across multiple manipulation tasks (Mandlekar et al., 2021). The study emphasizes the importance of high-quality demonstrations and highlights the difficulties in balancing exploitation of these demonstrations with exploration of novel behaviors.

Preliminaries

Problem definition and environment: We formulate the robotic manipulation task as a finite-horizon Markov Decision Process (MDP), defined by the tuple $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{T}, r, \gamma, \rho_0)$, where $\mathscr{S}$ and $\mathscr{A}$ are state and action space, respectively, $\mathscr{T}(s' | s, a)$ is the transition function, and $r(s, a)$ is the reward function.

Sparse reward structure: The reward function $r(s, a)$ in this environment is sparse, meaning that the agent receives a reward only when specific task goal is achieved.

$$r(s, a) = \begin{cases} 1 & \text{if } s \in \mathscr{G}, \\ 0 & \text{otherwise.} \end{cases}$$

Where $\mathscr{G}$ represent the goal distribution of the environment, and each goal $g \in \mathscr{G}$ corresponds to a particular configuration of kitchen objects or task completion state. Sparse rewards present a significant challenge as the agent must explore the state space efficiently to encounter rewarding states.

State Space $\mathscr{S}$: high-dimensional continuous space that captures the physical properties of the 9-DoF robotic arm and kitchen environment. Here we observe the,

1. *Robot State*
   a. Joint Positions $\mathbf{q} \in \mathbb{R}^9$, of the robot's joints.
   b. Joint Velocities $\dot{\mathbf{q}} \in \mathbb{R}^9$ of the robot's joints.
   c. End-Effector Pose ($\mathbf{p}_{\text{ee}} \in \mathbb{R}^3$,$\mathbf{R}_{\text{ee}} \in SO(3)$ *quaternion*) $\in \mathbb{R}^7$, the position and orientation of robot's end-effector.
2. *Object State*
   a. Positions $\mathbf{x}_{\text{obj}} \in \mathbb{R}^3$, of manipulable objects.
   b. Velocities $\dot{\mathbf{x}}_{\text{obj}} \in \mathbb{R}^3$, of these objects.
3. *Environment State*
   a. Task status $\in \mathbb{R}^1$, that is, it indicates the current state of task to be completed.

The full state vector in the code is represented as,

$$\mathbf{s} = [\mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}_{ee}, \mathbf{R}_{ee}, \mathbf{x}_{obj}, \dot{\mathbf{x}}_{obj}, \textbf{achieved\_goal}]$$

Overall, for all four manipulable objects, the state space dimensions are, $s \in \mathbb{R}^{50}$.

Action Space $\mathscr{A}$: continuous space that specifies the control inputs to the robot. Here, we control the,

1. *Joint Velocity Commands*
   a. $\mathbf{a} \in \mathbb{R}^9$ corresponding to each joint's angular velocity command.
2. *Gripper Control Command* (open/close) $\mathbf{a} \in \mathbb{R}$

Therefore, the action vector can be written as,

$$a = [\dot{\mathbf{q}}_{\text{cmd}}, g_{\text{cmd}}] \in \mathbb{R}^{10}$$

Transition Function $\mathscr{T}(s' \mid s, a)$ models the physics of the kitchen, that is the rigid body dynamics, and object interactions. Due to the complexity of the environment, $\mathscr{T}(s' \mid s, a)$ cannot be known analytically, but it is provided implicitly through the environment's simulator.

Objective function $J(\pi)$: To solve this MDP, we employ the SAC algorithm designed for continuous action spaces. SAC optimizes a stochastic policy $\pi_\phi(a \mid s)$ by maximizing the entropy-regularized objective,

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathscr{H}(\pi(\cdot \mid s_t)) \right) \right]$$

Here, $\mathscr{H}(\pi(\cdot \mid s)) = -\mathbb{E}_{a\sim\pi(\cdot|s)}[\log \pi(a \mid s)]$ is the entropy of the policy, and α is a temperature parameter that determines the trade-off between exploration (entropy) and exploitation (reward maximization).

Three-phase Softer Actor-Critic

In this section, we describe our proposed approach to implementing SAC algorithm, which leverages human

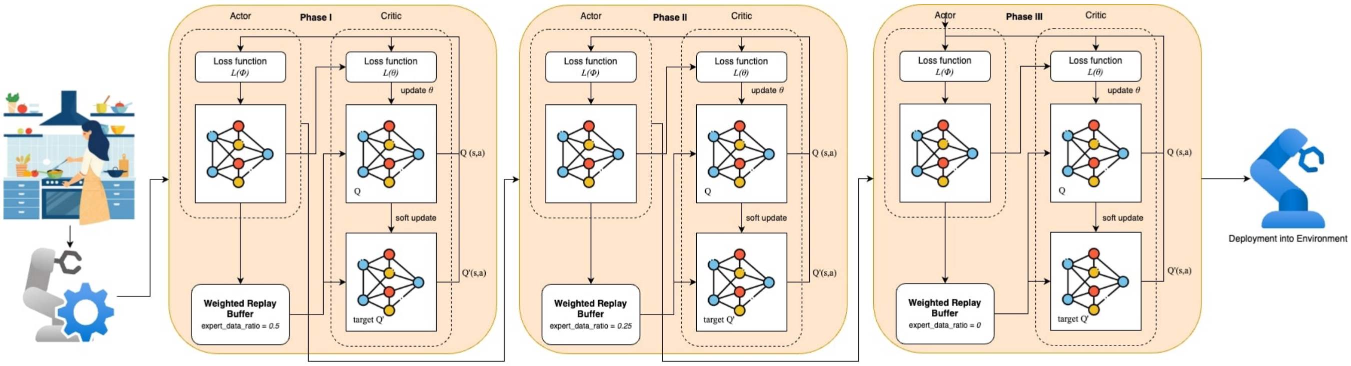

Figure 2. Three Phase Softer Actor-Critic Architecture

demonstrations ($\mathscr{D}$) and reinforcement learning to solve challenging real-world tasks in simulation environment. Our approach consists of three-phases, where in each phase we drastically reduce the reliance on $\mathscr{D}$, and enforce exploration while discouraging exploitation. The implemented architecture is shown in Figure 1.

Another reason for doing so is, we do not want this to become a supervised-learning problem where the agent is given a dataset and it has to learn its behavior. Although behavior-cloning refers to that ideology in some literature, we wanted the agent to explore enough to find a policy which is unique from the buffer present in $\mathscr{D}$. Note that, we do not expect the agent to do better, or outperform human in any manner at this point. However, we do hope that the agent does something or learns some policy which eventually converges to optimal (or near-optimal) performance. This hypothesis is proven to be true in the Results section of this paper.

Soft Actor-Critic

The key advantage of SAC is the the entropy regularization. This algorithm was chosen because here the policy is trained to maximize expected return and entropy tradeoff. This concept is closely related to the exploration-exploitation trade-off: an increase in entropy promotes greater exploration, potentially accelerating learning in subsequent stages. Additionally, it helps to mitigate the risk of the policy prematurely converging to a suboptimal local solution.

In entropy-regularized reinforcement learning, the agent is provided with an additional reward at each time step, which is proportional to the entropy of the policy at that specific time step.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\cdot\,|\,s_t))\right)\right]$$

Where α is the tradeoff coefficient.

In the challenging domain of robotic manipulation long-horizon tasks with sparse rewards, the entropy regularization inherent in SAC provides a robust framework for overcoming exploration challenges. By encouraging stochasticity, the algorithm facilitates the discovering task-relevant states and actions that may be difficult to identify in deterministic settings.

Dynamic adaptation of α

For instance, in our three-phase SAC framework, the entropy bonus becomes instrumental during the initial phases of learning, where the agent relies on demonstration data and must simultaneously explore its environment. By maintaining a high entropy, the agent avoids overfitting to demonstration trajectories and explores novel behaviors that could lead to improved performance. As the training progresses, the gradual reduction in reliance on demonstrations allows the agent to leverage its exploratory experience to refine its policy, converging to near-optimal performance.

Weighted Replay Buffer

In our proposed framework, the weighted replay buffer plays a crucial role in balancing the integration of expert demonstration data with the agent's self-learned experiences. This subsection highlights its key features and significance.

During the initial phase of training, the replay buffer includes a substantial proportion of expert demonstration data to bootstrap the learning process. This is controlled by the `expert_data_ratio` parameter, which specifies the fraction of the mini-batch sampled from expert data. As the training progresses, the ratio is gradually reduced, promoting transition from demonstration-guided learning to autonomous policy refinement.

The buffer supports randomized sampling of experiences, with optional prioritization of expert data. When sampling a mini-batch, a subset of the batch is drawn from the expert data, while the rest is sampled from the agent's own experiences. This ensures that both sources contribute to training in a balanced manner.

The replay buffer operates on a cyclic memory structure, ensuring efficient usage of memory resources by overwriting older experiences when the buffer is full. This allows the system to retain a balanced mix of recent and past experiences.

Workflow integration

In the workflow (as shown in Figure 1), the weighted replay buffer operates seamlessly within the training pipeline

1. *Phase I*
   - The buffer prioritizes expert data (*50% of mini-batch samples*) to guide the agent during early exploration.
   - The agent's policy is initialized with randomized actions to maintain entropy and prevent overfitting to demonstration data.
2. *Phase II*
   - The reliance on expert data is reduced to *25%*.
   - allowing the agent to leverage its own experiences more heavily while still benefiting from the structured guidance of demonstrations.
3. *Phase III*
   - The buffer fully transitions to sampling only agent-generated experiences.
   - Enabling the agent to rely solely on its learned policy.

Experiments

To evaluate the performance of our proposed three-phase Soft Actor-Critic (SAC) framework, we conducted a series of experiments in the simulated Franka Kitchen environment. The experiments compare the following methods:

1. *Vanilla SAC without Human Demonstrations*
   The agent learns solely through reinforcement learning, starting with a randomly initialized policy and relying entirely on sparse rewards from the environment.
2. *Vanilla SAC with Human Demonstrations*
   Human demonstration data is provided to bootstrap learning, and the agent exclusively uses expert data to sample experiences during all training episodes.
3. *Three-Phase SAC with Human Demonstrations*
   Our method integrates human demonstrations in a structured, multi-phase manner by progressively reducing the reliance on expert data over the training process. This encourages the agent to transition from demonstration-guided learning to autonomous exploration and policy refinement.

Our experiment aims to answer the following questions: (1) Does Three-phase SAC along with BC improve imitation learning as compared to Vanilla SAC that involves SAC? (2) Is Three-phase SAC more robust to challenging environments as compared to traditional methods which do not involve dataset? Videos and further experimental details are available at `https://keivalya.github.io/arm-manipulation-behavior-cloning/`

Experiment Setup

The experimental setup utilizes a 9-degrees-of-freedom *Franka* robot positioned within a simulated kitchen environment modeled in MuJoKo simulator, containing various household objects, including a microwave, kettle, overhead light, cabinets, and an oven. This environment is designed for multitask goal scenarios, requiring the robot to interact with these objects to achieve a specified target configuration.

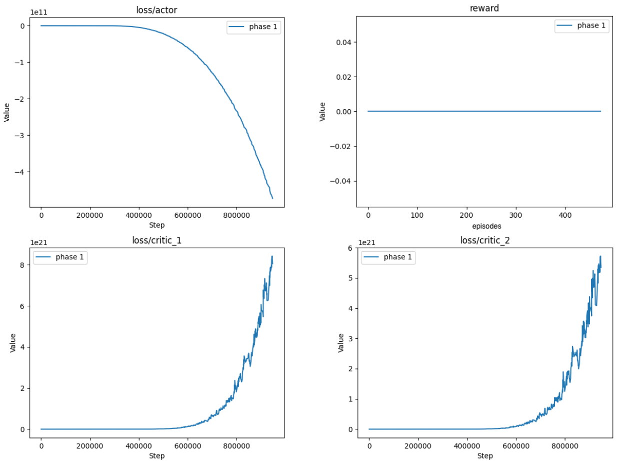

Figure 3. Vanilla SAC without dataset, experiment shows the results from completing ‘microwave opening’ task

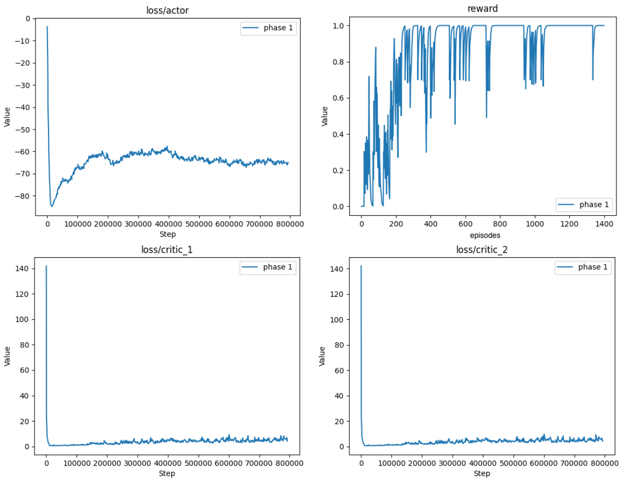

Figure 4. Vanilla SAC using dataset, experiment shows the results from completing ‘microwave opening’ task

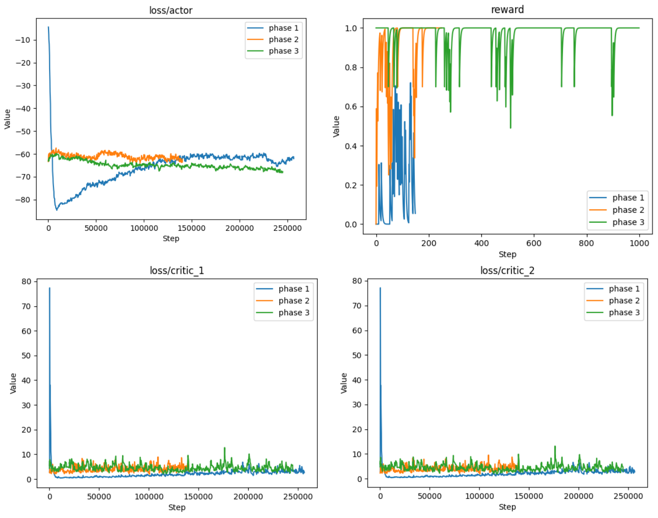

Figure 5. Proposed method (three-phase SAC) using dataset, experiment shows the results from completing ‘microwave opening’ task

Evaluation and Comparisons

The following metrics were recorded during the training:

1. *Reward per episode*, cumulative reward achieved by the agent in each episode, i.e. task successful/failed.
2. *Critic*$_1$ ($\mathscr{L}_{critic_1}$) *and Critic*$_2$ *Loss* ($\mathscr{L}_{critic_2}$), the mean squared error (MSE) between the predicted Q-values and the target Q-values for the two critic networks.
3. *Actor Loss* ($\mathscr{L}_{actor}$), the loss computed from the policy gradient.

Results

Vanilla SAC without $\mathscr{D}$*:* The agent struggles to explore the state-action space effectively due to the sparse reward structure and high-dimensional environment. Critic$_1$ and Critic$_2$ losses diverge rapidly as the training progresses, reflecting instability in the value estimation process. The agent fails to achieve meaningful rewards even after a large number of episodes, demonstrating the limitations of vanilla SAC in sparse reward settings, as shown in Figure 3. The lack of initial guidance or structured exploration results in poor learning and failure to converge to an optimal policy.

Vanilla SAC with $\mathscr{D}$*:* By relying entirely on expert data, the agent quickly starts achieving rewards in early episodes. However, the agent becomes over-reliant on the demonstration data and fails to explore sufficiently, leading to convergence to a sub-optimal policy. Critic losses remain relatively stable but fail to improve significantly, and the actor loss plateaus early, reflecting limited policy optimization, as shown in Figure 4. While demonstration data accelerates early-stage learning, the fixed reliance on expert data prevents the agent from adapting to novel scenarios or discovering optimal strategies.

Three-phase SAC: The agent benefits from human demonstrations during Phase I, gaining rewards (unlike Vanilla SAC without $\mathscr{D}$). As the reliance on expert data is reduced in Phases II and III, the agent transitions to exploring its environment autonomously, leveraging its learned experiences to refine its policy. Critic$_1$ and Critic$_2$ losses decrease steadily across training and the actor loss initially increases (due to entropy-driven exploration) but converges as the policy improves, indicating effective policy optimization, as shown in Figure 5. The agent achieves the optimal policy within a reasonable number of episodes, outperforming both baseline methods. The structured interplay between demonstration data and exploration allows the agent to achieve robust performance and discover the optimal policy efficiently.

Experiments on other tasks

Apart from microwave, we have experimented with solving other tasks too, such as switching on the top-left burner (Figure 6), opening hinge cabinet (Figure 7), and sliding slide-cabinet (Figure 8).

In order to analyze the robustness of our proposed method, we also train on various other tasks in the same environment. As a result we find that the ‘slide-cabinet’ task is easily learnt by the robot, that too from the Phase I itself. Upon further investigation it was found that this task was initially learnt by doing random movements, and by sheer luck the robot was able to swing its arm straight up, and perhaps accidentally open the cabinet to receive re-

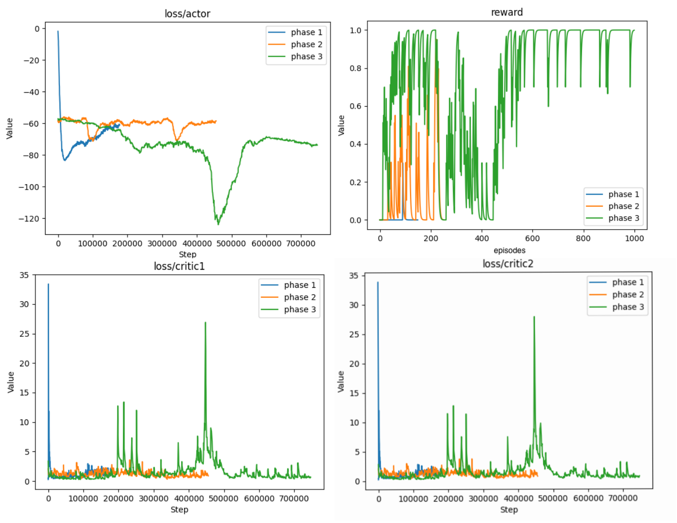

Figure 6. Three-phase SAC using dataset, experiment shows the results from completing ‘Opening Top-left Burner’ task

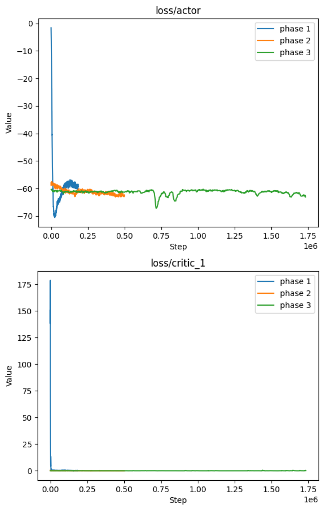

Figure 7. Three-phase SAC using dataset, experiment shows the results from completing ‘Opening Hinge Cabinet’ task

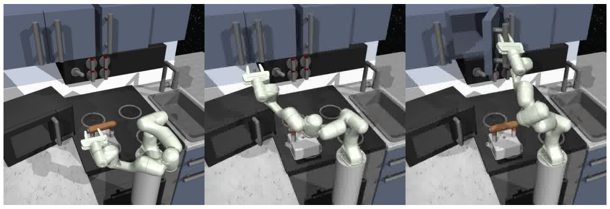

Figure 9. Example of opening hinge cabinet

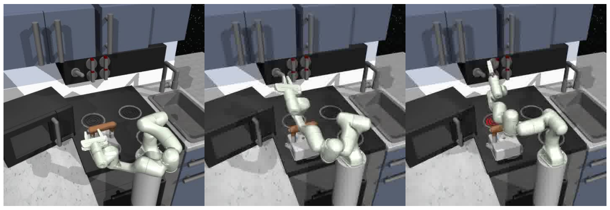

Figure 10. Example of turning on burner

ward. Then it slowly fine-tuned its policy to become more optimal and learn to perform it in satisfactory number of time steps.

On the other hand, the ‘hinge cabinet’ task seems to be the most difficult amongst all the tasks. On investigating it, we found that the dataset for hinge cabinet is full of noisy data, some of which do not even finish the task. However, even after giving such dataset, the agent learns an optimal policy which finishes the task successfully with the help of its intrinsic exploration capability.

Finally, the ‘top-burner’ task seems to be learnt in a reasonable amount of time with great precision. The robot learns to perform this task in optimal time-steps.

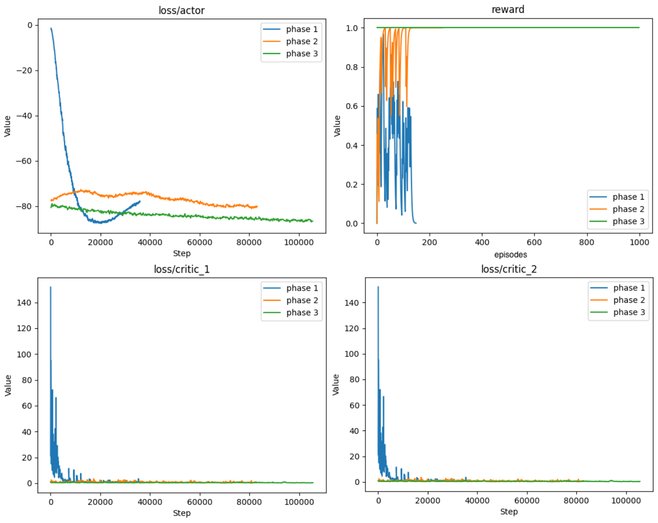

Figure 8. Three-phase SAC using dataset, experiment shows the results from completing ‘Opening Slide Cabinet’ task

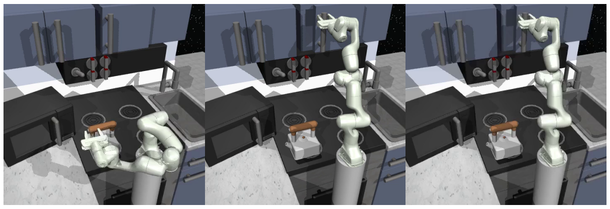

Figure 11. Example of opening slide cabinet

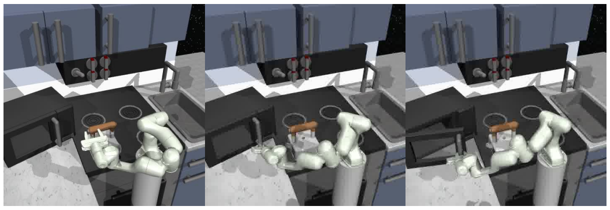

Figure 12. Example of opening microwave

Conclusion

In this paper, we introduced a three-phase Soft Actor-Critic (SAC) framework for solving continuous-state-space, and multi-stage robotic tasks. By leveraging human demonstrations in a structured and adaptive manner, our approach addresses the key challenges of sparse rewards and inefficient exploration commonly encountered in reinforcement learning for robotic manipulation tasks.

The proposed method incorporates a weighted replay buffer and a gradually reducing reliance on expert demonstration data. This design ensures that the agent benefits from initial guidance provided by high-quality demonstrations while progressively developing its own exploratory capabilities. The inclusion of entropy-regularized reinforcement learning further enhances the agent's ability to explore diverse trajectories and avoid premature convergence to suboptimal policies.

Experimental results demonstrate the effectiveness of our approach compared to two baseline methods: Vanilla SAC without human demonstrations and Vanilla SAC with fixed reliance on demonstrations. Our method achieved higher task success rates, greater sample efficiency, and stronger generalization capabilities. By dynamically balancing the trade-off between demonstration-based exploitation and autonomous exploration, the three-phase SAC framework consistently outperformed both baselines, particularly in the given environment.

This research highlights the potential of integrating expert data with reinforcement learning in a principled and adaptive way, paving the way for more efficient and robust learning in complex robotic tasks.

References

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction (2nd ed.).* The MIT Press.

Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. 2016. *Learning dexterous manipulation for a soft robotic hand from human demonstrations.* In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE Press, 3786–3793. https://doi.org/10.1109/IROS.2016.7759557

Ross, S., Gordon, G. &amp; Bagnell, D.. (2011). *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning.* Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 15:627-635 Available from https://proceedings.mlr.press/v15/ross11a.html.

Gupta, A., Kumar, V., Lynch, C., Levine, S. &amp; Hausman, K.. (2020). *Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning.* Proceedings of the Conference on Robot Learning, in Proceedings of Machine Learning Research 100:1025-1037 Available from https://proceedings.mlr.press/v100/gupta20a.html.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y. &amp; Martín-Martín, R.. (2022). *What Matters in Learning from Offline Human Demonstrations for Robot Manipulation.* Proceedings of the 5th Conference on Robot Learning, in Proceedings of Machine Learning Research 164:1678-1690 Available from https://proceedings.mlr.press/v164/mandlekar22a.html.